CSE535: Tic-Tac-Toe Game

Group 9: WeiSheng Chiu, Dipanshu Singh, Ujjwal Baranwal, Aashritha Machiraju, Geeth Nischal Gottimukkala, Devansh Tomar

***Abstract*—This report outlines the design and development of a Tic-Tac-Toe mobile application, built using Kotlin and Jetpack Compose, aimed at enhancing both user experience and gameplay intelligence. The game features three difficulty levels: Easy, Medium, and Hard. In Easy mode, the AI makes random moves to provide a relaxed experience, while Medium mode introduces a mix of random and optimized moves for balanced gameplay. The Hard mode employs the Minimax algorithm with alpha-beta pruning, delivering a highly challenging experience. The application boasts a clean, intuitive user interface and integrates key features such as game reset, winner detection, and persistent storage for tracking game history and player statistics. This report delves into the app's architecture, the AI's algorithmic strategies, and user experience design, demonstrating how modern mobile development can elevate a classic game.**

***Index Terms*—Android Development, Game Development, Mobile Game Development**

I. Introduction

Tic-Tac-Toe has long been a popular game due to its simplicity and strategic depth. In today's mobile-centric world, traditional games can be revitalized with advanced algorithms and intuitive interfaces to offer more engaging experiences. This project focuses on developing a Tic-Tac-Toe game for Android devices using Kotlin and Jetpack Compose, enhancing the classic game with modern AI and design principles.

The primary goal of this application is to deliver an enjoyable and challenging gameplay experience through three difficulty modes. Easy mode provides casual players with an AI that makes random moves, while Hard mode introduces a highly competitive AI that uses the Minimax algorithm with alpha-beta pruning to make optimal moves. Medium mode offers a balanced challenge by combining random and strategic moves, making it suitable for players seeking a moderate challenge.

Key features of the application include a responsive, dynamic game board, real-time move validation, a reset option, and persistent data storage to track game results, difficulty levels, and player performance. By integrating cutting-edge mobile development technologies like Jetpack Compose for UI design and SQLite/Room Database for persistent storage, the app ensures a seamless user experience. This project showcases how a classic game can be enhanced with modern development techniques, providing both nostalgia and innovation in one package.

II. Technical Approach

In this Tic-Tac-Toe implementation, the game features multiple difficulty levels where the AI moves are determined by the complexity of the algorithm. The "Hard" mode uses the Minimax algorithm with alpha-beta pruning to calculate optimal moves.

The AI uses the Minimax algorithm, enhanced with alpha-beta pruning, to make optimal moves by evaluating potential board states and simulating both its own and the player's responses. The goal is to maximize its advantage by selecting the best possible move to win or force a draw, while minimizing the player's chances of success. Alpha-beta pruning improves efficiency by skipping over board states that won't affect the final outcome, reducing the number of moves the AI needs to evaluate. The game's difficulty levels—Easy, Medium, and Hard—determine how the AI makes decisions, with Hard mode using optimal Minimax moves, Easy mode relying on randomness, and Medium mode blending both approaches. This combination allows the AI to provide a challenging and efficient gameplay experience.

A. Minimax Algorithm

The Minimax algorithm is a decision-making algorithm used for two-player games like Tic-Tac-Toe, where one player maximizes their score while the other player minimizes it. It assumes that both players play optimally, with the maximizing player, AI in this case, trying to get the highest score and the other the minimizing player, human player trying to minimize the opponent's score. In context of this game the Minimax algorithm helps determine the best move for the AI (player "O") by evaluating all possibe future board configurations and selecting the move that maximizes its chances of winning. The AI assumes that the human player will always make the best possible move to counter it.

The algorithm works by recursively simulating all potential moves from both the current player and the opponent, and then calculating a score for each possible board state. The score represents the desirability of that board state, with a higher score indicating a more favorable outcome for the AI and a lower score representing a better outcome for the player. The algorithm then selects the move that leads to the best score from the AI's perspective. The recursive structure of the algorithm explores all potential moves until it reaches a terminal state, where either a win, loss, or draw is determined.

- +10 if the AI wins
- -10 if the player wins
- 0 for a draw

From these terminal states, the algorithm backtracks and evaluates each move by comparing the outcomes. For the AI's turn, the algorithm chooses the move with the highest score, while for the player's turn, it chooses the move with the lowest score. By doing this recursively, the algorithm identifies the optimal strategy for the AI, effectively "thinking ahead" several steps into the game.

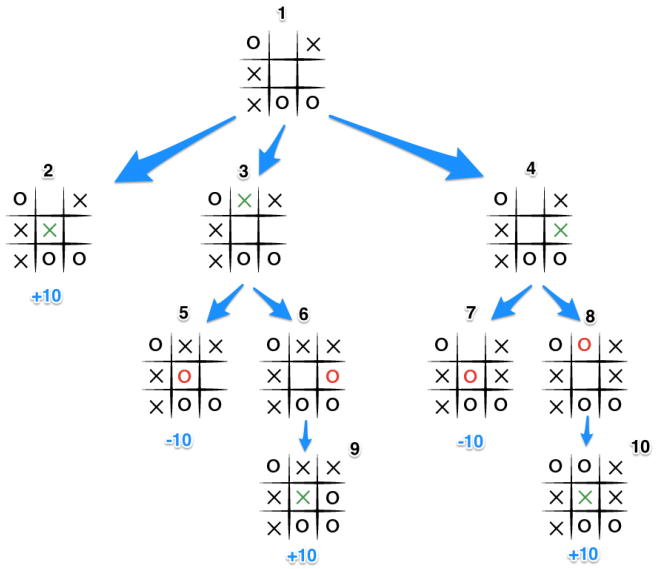


Fig. 1. Minimax visualization [Fox21]

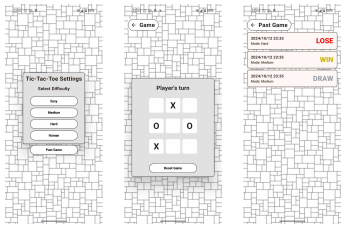


Fig. 2. Shows the screens of the application. On the left is the settings screen. In the middle is the game screen. On the right is the past game screen.

B. Alpha-Beta Pruning

Alpha-beta pruning is an optimization technique used in conjunction with the Minimax algorithm to reduce the number of nodes that need to be evaluated. Without alpha-beta pruning, the Minimax algorithm explores all possible moves and future game states, leading to high computational costs, especially as the depth of the game tree increases. Alpha-beta pruning significantly improves the efficiency of the Minimax algorithm by eliminating the need to explore branches of the game tree that cannot possibly influence the final decision. The main idea behind alpha-beta pruning is to keep track of two values, alpha and beta, which represent the best possible scores that the maximizer and minimizer can respectively achieve.

III. Design Choice

In this game application, there are three pages: The game screen, the settings screen, and the past game screen. Fig. 2 shows the screen virtually. When the user opens the game application, the settings screen lists the game mode, easy, medium, hard, and human. In addition to the game mode, the past game choice lets the user see the results of old games. This UI design gives the user a straightforward way to start a game on this screen. The game mode button is distributed vertically, so the small-screen phone could have a similar user experience.

After the user selects a mode, he/she will be led to the game screen. The game starts when the user takes a move on the game board. During the game, the user could press the back button on the left-top of the screen to go back to the settings screen and change the game mode. The game board is set on the middle part of the screen. There is no other thing lying above or below the game board. Therefore, the user could pay more attention to the game. Since the user can only select the game mode in the settings screen, there are fewer choices that the user needs to make on the game screen. This design strategy could give the user a stressless environment.

In the past game screen, it lists the records with a card. Each card represents an old game. The card starts with the game details: the time and the mode. The tail of the card shows the result of the game. There are three possible results: win, lose, and draw. We use font colors to distinguish those results visually. The winning game uses a gold color. This gives the user a happy emotion, just like winning the Olympic gold medal. The losing game uses a red color. This gives the user a nervous emotion, so they would be stimulated by their winning ambition. The stone color is set to display the drawing game. This would not provide any emotion when the user sees it, just like the meaning of the draw game.

We keep our design language the same on each screen. We chose an image that contains many lines as our background. This image looks simple and echoes the game since the tic-tac-toe is to occupy the cells to make a line. However, the dense lines make it hard for the user to distinguish the background and the content visually, so we package the content with a gray box. Except for the settings screen, the distribution of the elements is similar in the other two screens. The title and the back button are on the top of the screen, and the content is in the middle of the rest of the space. This unification eases the user from learning how to use the application since they only need to learn it once rather than learn a new control method for each screen. Since the phone screen is not like the computer screen, most of the elements are aligned in a vertical way.

IV. Implications and Challenges

In this project, the implications are closely tied to the knowledge gained and the technical skills developed. This project helped us deepen our understanding of mobile app development, particularly in using and utilizing different components of Android Studio in order to successfully build apps and games alike. We also developed an in-depth understanding of UI elements and backend AI elements through Jetpack Compose and implemented Minimax with alpha-beta pruning algorithms. Working with the Minimax algorithm, enhanced by alpha-beta pruning, gave us firsthand experience in optimization and decision-making processes in games. Additionally, the project highlighted the importance of creating efficient and user-friendly interfaces in mobile development. We also gained insights into how AI adapts to different difficulty levels, allowing for both random and optimal moves, which made the game more engaging and challenging for users. Lastly, using GitHub enhanced our understanding of collaborative

development, making version control and teamwork more efficient and organized.

However, our group faced several challenges throughout the project. One of the primary difficulties was the implementation of the Minimax algorithm with alpha-beta pruning, as balancing computational efficiency with game performance was tricky. This algorithm, although theoretically sound, required extensive debugging and testing to ensure that it worked correctly within the limited resources of mobile devices. Additionally, the need to implement the AI across varying difficulty levels (easy, medium, and hard) added complexity, especially in maintaining the fluidity of gameplay. Another challenge we faced was the integration of Jetpack Compose for the UI elements, especially since it was our first time working extensively with this framework. Designing a responsive and visually appealing interface while maintaining game functionality required a steep learning curve. Implementing a tiled repeated background added complexity, as we had to ensure that the background rendered correctly on various screen sizes without affecting the performance of the game. Additionally, time management and task delegation among group members were essential but often difficult to coordinate, especially when contributions varied in scope. Despite dividing tasks, some aspects of the project, such as debugging and fine-tuning the UI, required more collective effort than anticipated. These challenges, though demanding, ultimately taught us valuable lessons in teamwork, troubleshooting, and mobile development, which will benefit us in future endeavors.

V. Links

1) YouTube Video (Unlisted): CSE535 F24 Project 2: Tic-Tac-Toe (Group-9)
2) GitHub Repository (Private): Tic-Tac-Toe
3) Team Effectiveness Report: Team Effectiveness Report

References

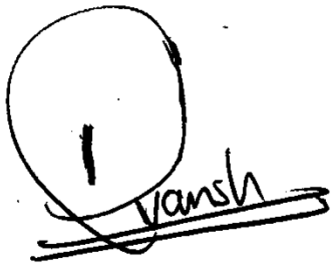
[Fox21] Jason Fox. *Tic Tac Toe: Understanding the Minimax Algorithm — Never Stop Building - Crafting Wood with Japanese Techniques*. Nov. 10, 2021. URL: https://www.neverstopbuilding.com/blog/minimax.

Team Effectiveness Report

Name	ASU ID	ASU Email Address	Contribution rate (%)
WeiSheng Chiu	1229560763	wchiu6@asu.edu	16.66%
Ujjwal Baranwal	1220406356	ubaranwa@asu.edu	16.66%
Dipanshu Singh	1220267958	dsingh47@asu.edu	16.66%
Aashritha Machiraju	1220510330	Amachira@asu.edu	16.66%
Geeth Nischal Gottimukkala	1219532899	ggottimu@asu.edu	16.66%
Devansh Tomar	1220103989	dtomar1@asu.edu	16.66%

WeiSheng's Signature: 

Ujjwal's Signature: 

Devansh's Signature: 

Dipanshu Signature:

Dipanshu Singh.

Aashritha’s Signature

Geeth’s Signature: